4-2013

Strong Location Privacy: A Case Study on Shortest Path Queries [Invited Paper]

Kyriakos MOURATIDIS
Singapore Management University, kyriakos@smu.edu.sg

DOI: https://doi.org/10.1109/ICDEW.2013.6547441

Strong Location Privacy: A Case Study on Shortest Path Queries

[Invited Paper]

Kyriakos Mouratidis

School of Information Systems
Singapore Management University
kyriakos@smu.edu.sg

***Abstract*— The last few years have witnessed an increasing availability of location-based services (LBSs). Although particularly useful, such services raise serious privacy concerns. For example, exposing to a (potentially untrusted) LBS the client's position may reveal personal information, such as social habits, health condition, shopping preferences, lifestyle choices, etc. There is a large body of work on protecting the location privacy of the clients. In this paper, we focus on shortest path queries, describe a framework based on *private information retrieval* (PIR), and conclude with open questions about the practicality of PIR and other location privacy approaches.**

I. Introduction

The wide availability of positioning systems and the diffusion of smart-phones has led to an expanding market of location-based services (LBSs). Clients of these services may use their mobile devices to get driving directions to their destination, to retrieve facilities close to their location (e.g., clinics, pharmacies, police stations), to learn who of their social contacts are nearby, etc.

Practical as these services may be, users view them with increasing skepticism. The very nature of the queries may disclose personal information (such as health status, shopping habits, lifestyle choices, etc) which may be tracked and misused by the LBS. Possible forms of misuse include commercial profiling, unsolicited and intrusive advertising, etc. The recent example of a leading mobile device company, which had been tracking the locations of its clients without their consent [1], [24], underlines the serious privacy risks in using LBSs.

Two general approaches have been taken so far to hide the location of the querying client from the LBS; location obfuscation, and PIR-based methods. The first category includes schemes that replace the client's position with a nearby location [28], with a spatial region that encloses it [14], [8], or with a set of candidate locations (including the client's actual position) [4], [11]. Location obfuscation is said to offer *weak* privacy, in the sense that the LBS always learns some (albeit not exact) information about the user location.

The second class of methods relies on private information retrieval (PIR) [2]. PIR is a primitive that allows a data item (e.g., a disk page) to be retrieved from a server, without the server obtaining any clues about which item was retrieved. Unlike obfuscation, PIR offers cryptographic privacy guarantees, based on reductions to problems that are either computationally infeasible or theoretically impossible to solve. In this sense, PIR-based methods offer *strong* location privacy (e.g., [17], [10]). The downside of these techniques is that PIR is generally resource-intensive. Although recent PIR protocols achieve reasonable response times (in the order of seconds over Gigabyte databases [25]), incorporating them into spatial query processing is not straightforward nor always practical.

In this paper, we use shortest path queries in transportation networks as an example to showcase issues, challenges and performance implications of the two paradigms, with a focus on the PIR approach. A transportation network could represent the road segments in a city, where each segment is associated with a cost (e.g., its length or the time required to drive through it). The query computes the sequence of road segments to reach from a source s (usually the client's current location) to a destination t so that the summed cost along the path is minimized. This is one of the most common queries in LBSs. Examples of popular services that support shortest path computation include Google Maps, Map Quest, etc. Note that, unlike range and nearest neighbor queries, shortest path computation may disclose information not only about the current position of the client, but also about her intended destination and path taken.

The paper is structured as follows. Section II surveys existing location privacy techniques, including two obfuscation methods for shortest path queries. Section III outlines a PIR-based methodology that offers strong privacy, i.e., where the LBS answers shortest path queries without deducing any information about them. Section IV sketches two techniques that implement this methodology. Section V presents representative performance results. Section VI concludes the paper with thoughts about the practicality of location privacy approaches, and with open questions that future research should consider.

II. Overview of Existing Work

A. Obfuscation Methods

Spatial k-anonymity is a type of obfuscation for location privacy that is inspired by the concept of k-anonymity in relational databases [21]. The architecture includes (i) the clients, (ii) the LBS that hosts a spatial database and answers

queries on it, and (iii) a trusted mediator, commonly referred to as the *Anonymizer*. The clients update the Anonymizer about their most recent locations, and forward to it their queries. Posed a spatial query, the Anonymizer replaces the coordinates of the originating client u with a region (usually a square or a circle) that includes u and at least $k-1$ other clients. This k-anonymous region is forwarded to the LBS, which reports back to the Anonymizer possible query answers for any point inside the region. The Anonymizer filters the results, and forwards to u the actual answer to its query. The privacy assurance offered to clients is that, even if the LBS knows the exact locations of all clients, it is unable to identify which among the k clients inside the anonymous region is the query originator.

There exist several spatial k-anonymity methods for range and nearest neighbor (NN) queries in Euclidean space [14], [8], as well as adaptations that drop the Anonymizer from the model, and instead have the clients collaboratively form the k-anonymous regions [3], [7]. There also exist spatial k-anonymity methods for NN processing on road networks [23], [15]; here, instead of a spatial region, a set of road segments is used to anonymize u, with the requirement that at least $k-1$ other clients are also located on these segments.

Another class of obfuscation methods use fake locations instead of k-anonymous regions. In [4], [11], for instance, the client forwards to the LBS a set of fake query locations along with her actual position. The assumption is that the LBS is unaware of clients' locations, so that it is unable to tell apart the decoys. Another approach is to choose a fake location u' near the client and forward it as the query point [28], [18]. In this setting, nearest neighbor queries can be answered by incrementally fetching from the LBS the NNs of u' and stopping when the set of retrieved data objects is guaranteed to contain the NNs of the actual client's location.

There are two obfuscation methods to protect shortest path queries in road networks, [12] and [22]. The former assumes the existence of an *Obfuscator*, which plays a role similar to the Anonymizer, i.e., it serves as a trusted mediator between the clients and the LBS. A client u querying about the shortest path from a source s to a destination t, relays its request to the Obfuscator. The Obfuscator appends s and t with a number of decoys, producing obfuscation sets S and T, which it then forwards to the LBS. The LBS computes all shortest paths from any candidate source in S to any candidate destination in T. Upon receipt of these paths, the Obfuscator picks the one that corresponds to the real source and destination, and reports it to the client. To improve performance, [12] suggests choosing decoys close to the real s and t, respectively.

On the other hand, [22] replaces the source and destination with their containing regions. Unaware of the exact s and t locations, the LBS computes multiple shortest paths between the two regions. To achieve reasonable performance, [22] presents an approximate scheme, i.e., the reported path has a bounded deviation in distance units from the shortest path.

By definition, obfuscation methods disclose some information about the query location, thus providing weak privacy. For instance, spatial k-anonymity methods reveal to the LBS that the user lies inside the k-anonymous region, which is only a part of the entire data space (and usually a small one). Similarly, in methods that append u with decoys, the LBS is offered a finite set of alternatives for u to be located at. If the LBS can additionally disqualify some decoys (e.g., via contextual knowledge), its chances of guessing correctly the client's location increase. On the other hand, in [28], [18] the LBS does not acquire the actual client location, but it still gains information about her whereabouts, because u' lies in her vicinity.

For the specific case of shortest path privacy, in [12] the LBS obtains knowledge of a finite set of alternatives for s and t ($|S|$ and $|T|$ candidate locations, respectively) which, moreover, lie near the actual source and destination, providing a rough idea of their positions. [22], on the other hand, reveals the regions that contain the source and destination.

Although not an obfuscation method per se, another approach considered for location privacy is space transformation [9], [26], [27]. In this model, the database owner, who is different from the LBS, maps the data from the original Euclidean space into a transformed space using a keyed function. A querying client u (in possession of the secret key) converts her location into the transformed space and forwards it to the LBS. The latter, although unaware of the secret key and thus unable to map the data and query back to the original space, is still able to compute the query result. [9] supports approximate NN processing, [26] provides exact NN results, and [27] additionally answers range queries. Transformation techniques are meant for single client settings, because possession of the secret key by multiple ones implies that any client may collude with the LBS to "decrypt" another's query. Also, transformation schemes are susceptible to access pattern attacks [25]. For example, the LBS may observe the access frequencies of items in the transformed space and use them in tandem with contextual knowledge about the original space to deduce a (partial) mapping between the two spaces.

B. PIR-based Methods

Private information retrieval (PIR) is a primitive for retrieving data hosted by a server, without the server learning anything about the clients' access patterns [2]. The privacy guarantees of PIR protocols rely on reductions to problems that are either computationally infeasible or theoretically impossible to solve (for a complete survey of PIR techniques and an in-depth description of their inner workings, the interested reader is referred to [5]).

Many types of single-server PIR are known to incur prohibitive computation and/or communication overheads for sizable datasets [20]. However, recent *hardware-aided* PIR protocols are shown to be more practical. These protocols utilize a tamper-resistant, *secure co-processor* (SCP) that is installed at the server and is trusted by the clients. For example, [25] features constant communication and amortized polylogarithmic computation cost.

PIR schemes have been applied in the context of spatial queries. The first such method appeared in [6] for NN pro-

cessing, but relied on a particularly expensive PIR protocol. Subsequent proposals utilize hardware-aided PIR and report more reasonable computation/communication overheads for NN retrieval (a few seconds for Gigabyte databases) [10], [17]. Importantly, [17] asserts that it is not enough to retrieve disk pages from the LBS via a PIR protocol, but the number of pages accessed should be the same for all queries. Otherwise, clues may be given about the data of interest and therefore about the query itself. As a case study, in the following we describe a recent PIR-based framework for private shortest path queries [16].

III. A PIR-based Methodology for Shortest Paths

In this section, we frame the problem, define the privacy objective and outline a provably secure methodology for private shortest path computation. We then present characteristics of SCP technology and of the employed PIR protocol that guide solution design.

A. Problem Formulation and System Model

Query: A road network is modeled as a weighted graph $G = (V, E)$, where V is the set of nodes, and E the set of edges. The nodes $v \in V$ represent junctions, or positions on a road where the traffic conditions or the orientation change, such as road turns. Every edge $e \in E$ connects two nodes and is associated with a positive weight $w(e)$ that models the cost to traverse e, e.g., the traveling time from one node to the other, the length of e, etc. A path from a source node $s \in V$ to a destination node $t \in V$ is a sequence of edges starting at s and leading to t. The cost of a path is defined as the sum of costs across its edges. The path from s to t with the smallest cost is called the *shortest path* and is denoted as $SP(s, t)$. E is assumed to include directed edges and s, t to lie on two network nodes. The discussion easily extends to undirected edges and s, t that lie anywhere on the road network. All nodes have Euclidean coordinates associated with them.

Architecture: The road network G is hosted by an LBS – G may be owned by the LBS itself or another entity. The LBS stores on the disk the graph data and any indexing information thereof, organized in equal-sized blocks (pages). The clients of the LBS pose shortest path queries on G, and the LBS needs to report the results back to them. A secure co-processor (SCP) is installed at the LBS, and offers a PIR interface for clients to retrieve disk pages from the database of the LBS. Details about the SCP and the PIR protocol employed are given in Section III-B. Although we assume that the database resides on disk, the PIR interface (and the entire framework presented) applies to storage in main memory or a solid state drive.

The architecture is visualized in Figure 1. When a client wishes to pose a query, she establishes an encrypted connection (e.g., SSL) with the SCP and answers the query via a multi-round protocol. In each round, the client requests specific disk pages from the SCP, which retrieves them from the database (one by one) in a way oblivious to the LBS. The data fetched determine the page requests in the next round, and so on, until the shortest path is computed.

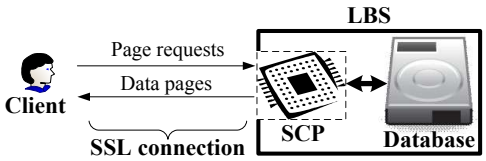

Fig. 1. System architecture

Adversary: The adversary in this model is the LBS. We assume that it knows the client's identity (e.g., via user log-in) or may infer it[1]. The adversary is curious, but not malicious [25], i.e., it wishes to gain information about the clients' queries, yet it executes page access routines correctly, and would not falsify the data in any way. The road network information and its index (if any) are not encrypted, i.e., their plaintext is available to the LBS, who may well be their owner. The adversary is also aware of the processing protocol in use. Its computational power is polynomially bounded (a common assumption that enables the use of cryptographic primitives, such as secure hash functions, etc).

Security Objective and Privacy Guarantee: The objective is practical protocols for processing shortest path queries at the LBS *without the latter deducing any information about the queries*. The database comprises a set of files, e.g., a header file, a graph data file, an index file, etc. Similar to [17], it is required that every shortest path query follows the same *query plan* – this is necessary in order to achieve the privacy goal, as will become clear in the security proof below. Specifically, we need to ensure that every query (i) executes in the same number of rounds, (ii) in each round it accesses the same files in the same order, and (iii) from each file accessed in a specific round, it retrieves the same number of pages. The query plan is determined by the processing protocol (we will see how) and is publicly available. For example, if the protocol suggests that in the second round 5 pages are fetched from file F_1 and then 10 from file F_2, every query in its second round must fetch 5 pages from F_1 followed by 10 from F_2 (in this order). This implies that even though a certain query may need fewer than the specified pages from a file, the protocol pads its requests with dummy page retrievals in order to conform to the query plan. The following theorem proves that this methodology achieves the security objective.

Theorem 1: The above methodology leaks no information to the adversary about the shortest path query. Equivalently, every processed query is indistinguishable from any other.

Proof: Each page requested from a file is retrieved via an established PIR protocol. Therefore, the adversary is oblivious of which page of the file is being read. What is only visible to the LBS is that a page is being accessed in the specific file. Since all queries follow the same query plan, the number of

[1]Even without user log-in (such as in Google Maps), identification is possible via background knowledge (e.g., user profile/search history), especially if information about the client's source and destination also leaks.

page retrievals in the various files and their chronological order is identical for all queries, lending the adversary no means to tell any two of them apart. For this reason too, even if the exact same query is re-executed, the LBS is unable to detect that it is processing the same query. Having established that the adversary gains no information from query execution, the proof is completed by the fact that it is also unable to intercept the client's page requests (to the SCP) and the page contents sent back from the SCP (to the client), because they are transmitted via a secure connection (SSL). ■

The general methodology described above fulfills our privacy objective. However, the challenge now lies in determining specific processing schemes which (i) ensure that all queries follow the same query plan, and (ii) are practical in terms of performance (e.g., in terms of query response time, space overhead, etc). Before presenting any schemes, we provide some background about hardware-aided PIR that determines the design principles.

B. Background and Design Considerations

A PIR interface is required to enable clients to securely access the database of the LBS – as explained in Section II-B, hardware-aided PIR is currently the only practical option. To provide a readily deployable framework, we rely on existing SCP technology and PIR protocols. Hence, we review their properties and limitations.

The SCP is trusted by the clients and installed at the processing server. It has access to the server's disk and may execute a set of cryptographic primitives. SCPs support complete tamper detection, so that clients may remotely assess whether they operate unmolested and unobserved by any potential adversary. The tamper-resistance of SCPs comes at the cost of excessive heat dissipation which, in turn, limits their computation speed and memory capacity. General purpose SCPs are available in the market, such as the IBM 4764 PCI-X Cryptographic Coprocessor.

To fetch disk pages obliviously from the database of the LBS, the protocol of [25] is employed, due to its superior performance (note however that alternative PIR protocols could be used). Retrieving a disk page has an amortized computation cost of $O(\log^2 N)$, where N is the total number of pages in the accessed file. The amortized complexity is used because some retrievals may involve reorganization in parts of the file. In absolute terms, a real implementation on IBM 4764 takes around one second to retrieve a page from a Gigabyte file. The communication cost incurred is constant, i.e., the amount of data transferred to the client (via the SSL connection) have the same size as the original disk page read.

The computation cost of the protocol, albeit much smaller than other PIR approaches, is still several times larger than a plain (unsecured) disk read. To ensure viability, *a key objective in solution design is to keep the number of pages fetched per query (i.e., per shortest path computation) as small as possible*. This will also limit the communication cost.

Importantly, the protocol of [25] requires that the SCP has at least $c \cdot \sqrt{N}$ memory, where c is a parameter with a typical value of 10. In conjunction with the limited memory on the SCP, this implies that files larger than a certain size cannot be supported – in the experiments we present later, the SCP (IBM 4764) has 32 MByte RAM and may support files up to 2.5 GByte. It is also indicative that the memory capacity in SCP technology increases much slower than, say, hard disk capacity. Therefore, *in the solution design it is essential to keep the database size small*.

A final remark regards the choice to adopt a multi-round methodology, i.e., to have the client lead query processing with repetitive page requests. One could wonder why the processing logic is not completely shipped to the SCP, so that it runs locally the necessary rounds of the protocol, and directly reports to the client the query result (shortest path). The reason is that programming on the SCP is particularly cumbersome, and also that complicated code may lead to prolonged execution due to the aforementioned overheating issues. Hence, the SCP is used merely as an interface to securely fetch specific disk pages (one at a time), using off-the-shelf functionality.

IV. PIR-based Privacy Schemes

Here, we present two processing schemes that follow the methodology described in Section III. As already established, the main performance factors are query processing cost and database size. Note that the former is linked directly to the maximum number of pages needed for any possible source-destination pair (due to the fixed query plan requirement).

A. Concise Index Scheme

The first scheme is termed *Concise Index* (CI). It features a minimal space overhead and a manageable query processing cost. In CI the database consists of four files, namely the *header*, the *look-up*, the *network index* and the *region data* file; we denote them as F_h, F_l, F_i, F_d, respectively. Their roles are as follows.

- *Header*: CI partitions the network into regions. The header helps the client map her source and destination to their host regions. It also includes the query plan.
- *Look-up*: It enables browsing the network index file.
- *Network index*: It includes pre-computed information that helps guide the shortest path search.
- *Region data*: It stores the actual network information of each region, i.e., node coordinates, adjacency lists, etc.

We first present the pre-processing steps in CI, i.e., network partitioning and pre-computation (Sections IV-A.1 and IV-A.2). Next, we describe the exact contents of each file (Section IV-A.3). Then, we discuss the derivation of the query plan and the query processing algorithm (Section IV-A.4).

1) Network Partitioning: CI, as well as subsequent schemes, relies on a partitioning of the road network into regions. The choice of partitioning method is important. One requirement is that it must be easily representable in terms of Euclidean coordinates. The reason is that clients are unaware

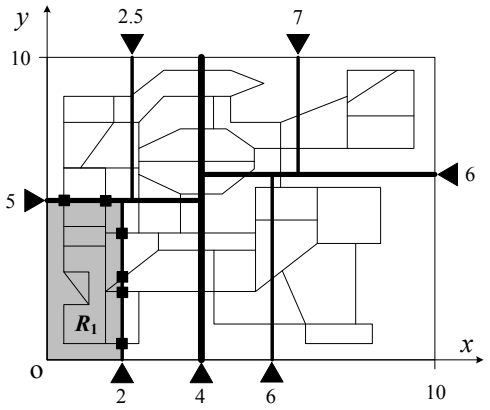

Fig. 2. KD-tree partitioning and border nodes

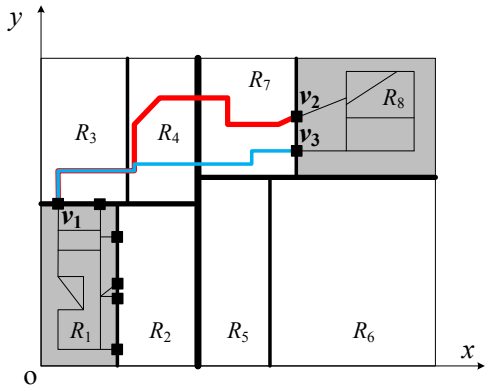

Fig. 3. Shortest paths between border nodes

of node or region identifiers[2], and may only express their source and destination in terms of Euclidean coordinates. Another requirement is that it should facilitate query processing, implying that regions should be chosen such that shortest paths are likely to cross as few of them as possible. Last but not least, the partitioning information should be expressible in a concise form, because it will be sent to the clients (as part of F_h) over a communication network.

A simple partitioning method is to superimpose a KD-tree (in Euclidean space) on the road network. This technique produces regions of comparable quality (in terms of facilitating shortest path computation) to more sophisticated alternatives [13]. Additionally, the tree structure (which essentially determines the mapping between Euclidean coordinates and network regions) can be represented in a very concise form.

Each leaf of the tree holds the nodes that lie inside its spatial extent; a node's information includes its identifier, its coordinates and its adjacency list (i.e., the list of adjacent nodes and the weights of the corresponding edges). Every leaf determines a region and is associated with a region identifier R_i. Figure 2 illustrates the KD-tree partitioning of a sample road network. The bold lines correspond to the split lines of the tree nodes. Region R_1 is defined by the leaf shown shaded, and holds the information of all nodes inside. The tree structure can be represented simply by the splitting coordinate (either on the x or y axis) used in every node of the tree.

The idea in CI is that node information for each region is placed in a single disk page. That is enforced by splitting tree nodes until the network information in each leaf fits in a page.

2) Pre-computation: CI pre-computes and materializes some shortest path information. Key in this process is the notion of *border nodes*. These are intersection points of the network edges with the splitting lines of the KD-tree. In Figure 2, for example, region R_1 has 6 border nodes, represented as solid squares. Border nodes are treated as normal network nodes during pre-processing, but they are discarded afterwards (i.e., not stored).

[2]Node and region identifiers are a matter of naming during database creation, and cannot be assumed known to the client in advance.

The fundamental property of border nodes is that any path starting from a source s inside some region R_s to a destination outside of it must pass through one of the border nodes of R_s. Similarly, any path to a destination t in region R_t (from a source outside of it) passes through a border node of R_t. Consider a shortest path $SP(s,t)$ and let v and v' be the border nodes of the source and destination region, respectively, that appear in this path. Due to its cost minimality, $SP(s,t)$ is guaranteed to include $SP(v,v')$. The above facts combined suggest that $SP(s,t)$ passes necessarily via $SP(v,v')$ for some border node pair (v,v'). In Figure 3, assume that s is somewhere in R_1 and t in R_8. If the shortest path $SP(s,t)$ passes through border node v_1, it necessarily includes either $SP(v_1,v_2)$ (shown red) or $SP(v_1,v_3)$ (shown blue), where v_2 and v_3 are the border nodes of R_8.

Based on this observation, CI computes for every pair of regions R_i, R_j the shortest paths from all border nodes in R_i to all border nodes in R_j. Let $S_{i,j}$ be the set of intermediate regions crossed by at least one of these paths. For example, the consideration of border node pair (v_1,v_2) in Figure 3 would include (the identifiers of) R_3, R_4, R_7 into region set $S_{1,8}$. By definition, any shortest path from a source in R_i to a destination in R_j may pass only through R_i, R_j and regions in $S_{i,j}$. This pre-computation process is also necessary for $S_{i,j}$ sets where $i=j$ (i.e., when source and destination regions are the same) because a shortest path between border nodes of R_i might still pass through a neighboring region.

3) File Formation: After partitioning and pre-computation, the four files are formed.

Region Data File (F_d): As mentioned previously, F_d includes exactly one page for every region R_i. Inside it keeps the network information of R_i, including node identifiers, their adjacency lists and incident edge weights.

Network Index File (F_i): F_i contains the pre-computed $S_{i,j}$ information. The region sets $S_{i,j}$ are stored into pages in ascending order of composite key (i,j). They are placed contiguously into pages, with the objective of minimizing the total number of pages each of them spans. In particular, for $S_{i,j}$ sets with size smaller than a page (as in the vast majority of cases), we prevent them from stretching over two pages.

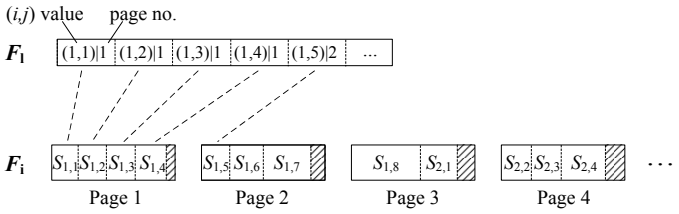

Fig. 4. F_l and F_i example

This implies that during file formation, if the free space in a page is not enough to host the next $S_{i,j}$ set (in (i,j) order), the space is left unutilized and the region set is placed in the next page of the file. This is important in order to reduce the PIR retrieval cost per region set. Figure 4 (in its lower part) illustrates an example of F_i. The striped space at the end of the pages is unutilized.

Look-up File (F_l): F_l is essentially a dense index over F_i, as shown in Figure 4. Specifically, for every (i,j) pair, F_l stores a look-up entry that indicates the page number in F_i that holds region set $S_{i,j}$. The F_l entries are sorted on composite key (i,j). The pages in F_l are packed, i.e., each stores the maximum possible number of look-up entries. This implies that for any pair (i,j), a division by that number indicates the F_l page that holds the corresponding look-up entry (which in turn leads to the actual $S_{i,j}$ data in F_i).

Header File (F_h): The header includes the KD-tree information that allows mapping s and t to their host regions. For each leaf of the KD-tree (i.e., for each region) the header also stores (i) a region identifier (e.g., R_1, R_2, etc), and (ii) the page number in F_d that holds the actual network information of the region. The header additionally specifies the query plan and meta-data about the other three files (e.g., filename, size, record length for F_l, etc). F_h is small and needs to be downloaded by every client who wishes to pose a query. Therefore, it discloses no information about the query itself, and is downloaded in full directly from the LBS, without involving the PIR interface.

4) Query Processing: Consider a client who wishes to know the shortest path from source s to destination t, and ignore the query plan for the time. In the first round of processing, the client receives the header file F_h. Based on the coordinates of s and t, she uses the KD-tree information to determine the source and destination regions R_s, R_t. Note that there is no requirement that s and t are network nodes; they could lie anywhere on the road network.

In the second round, the client uses the PIR interface and fetches the page in the look-up file F_l that corresponds to pair (s,t). She extracts the look-up entry for the specific pair and learns the page number in the network index file F_i that stores the $S_{s,t}$ set. In the third round, she fetches that page from F_i (via the SCP); if $S_{s,t}$ stretches to nearby pages, they are also retrieved.

In the fourth round, the client requests (via the SCP) the pages of F_d that include the network information of R_s, R_t, and all regions in $S_{s,t}$. Upon receipt of these data, she possesses a subgraph of G that is guaranteed to contain the desired shortest path. $SP(s,t)$ is computed using Dijkstra's algorithm in this subgraph.

Query Plan: In addition to the above accesses, the query plan may require extra (dummy) page retrievals. In the first round the entire F_h is downloaded. In round two, there is always a single page fetched from F_l. In round three, we force each query to retrieve as many pages from F_i as the maximum number of pages spanned by any $S_{i,j}$ set. This means that even if a single $S_{i,j}$ set spreads over three pages in the file (while every other fits in one or two), any query will need to make three retrievals in F_i. An important implementation detail here is that the client does not know in advance how many pages $S_{s,t}$ spans, but she knows from the query plan that the maximum it could be is three. Therefore, it requests for the page indicated by the look-up entry for pair (s,t) plus the subsequent two pages.

Regarding round four, let m be the maximum number of regions inside any $S_{i,j}$ set. Recall that each region's data fit in a single page of F_d. The query plan ensures that every query accesses $m+2$ pages in F_d (the extra two pages account for R_s and R_t).

B. Passage Index Scheme

As we will see in the experiments, CI requires little extra space compared to simply storing the raw network data. However, the longest paths in G may span a considerable number of regions (implying that value m, mentioned in the previous paragraph, may be large). That leads to a significant number of PIR accesses in F_d which dominate the response time. Motivated by this fact, the *Passage Index* (PI) scheme uses more space, but achieves a drastic reduction in the number of pages needed, and thus in response time.

In PI, instead of having the client retrieve all intermediate regions between R_s and R_t, we materialize an exact subgraph that links them. Specifically, pre-computation is the same as in CI. However, instead of keeping $S_{i,j}$, we record for every pair of R_i and R_j the exact edges that appear in one or more shortest paths between their border nodes. Essentially, these edges define a subgraph $G_{i,j}$, such that every shortest path from R_i to R_j is guaranteed to pass entirely through the union of R_i, R_j and $G_{i,j}$. In the example of Figure 3, $G_{1,8}$ includes, among others, the edges that belong to the two shortest paths (shown in red and blue).

PI involves four files, formed as explained in Section IV-A.3, the difference being that the network index file includes the $G_{i,j}$ information (instead of $S_{i,j}$). Placement into physical pages follows the same principles. In query processing, however, there are only three rounds. The first two are identical to CI, while the third fetches (i) from F_i the subgraph $G_{s,t}$ that corresponds to the source and destination regions, and (ii) the two pages in F_d that hold the network information of R_s and R_t. Regarding the query plan, let h be the maximum number of pages spanned by any subgraph $G_{i,j}$ in the network index file. Each shortest path computation should retrieve in the

first round the entire header (directly from the LBS, without involving the SCP), in the second round one page from F_l, while in the third round exactly h pages from F_i and two pages from F_d. Note that if $h = 1$, which may be the case for a small network, PI answers the query with only four PIR accesses.

In PI the network index file vastly dominates the space requirements. To reduce its size, the crucial observation is that subgraphs $G_{i,j}$ exhibit locality (i.e., for nearby (i, j) pairs the subgraphs share many common edges). This enables an in-page compression technique. When inserting $G_{i,j}$ into an F_i page, we chose as *reference* the subgraph in the same page with the largest number of common edges, and store as *delta information* the edges in $G_{i,j}$ that are missing from the reference subgraph.

V. Experiments

In this section, we provide representative empirical results on a real road network. In addition to the PIR-based methods (CI and PI), we include measurements for an obfuscation method based on [12] (OBF); this is for the sake of a performance indication only, because privacy-wise OBF leaks substantial information about the client queries, as explained in Section II-A.

All methods were implemented in C++ and executed on a machine with an Intel Core2 Quad CPU 2.83 GHz and 4 GByte of RAM. The evaluation uses the Argentina road network (with 85,287 nodes and 88,357 edges), obtained from the Digital Chart of the World[3]. The machine uses a Seagate 320 GB (7,200 RPM) hard disk,[4] with 11 ms disk seek time, 125 MBbyte/s disk read/write rate, and 4 KByte disk page size. The *IBM 4764 PCI-X Cryptographic Coprocessor*[5] is adopted as the SCP; its performance is simulated strictly. The SCP has 32 MByte memory and may support file sizes up to 2.5 GByte. Table I summarizes the specifications of the SCP and the hard disk (these values determine the time to retrieve a disk page via the PIR interface, as detailed in [25]). The client communicates with the LBS using a link with round trip time of 700ms and bandwidth 384 Kbit/s (i.e., 48 Kbyte/s) – this corresponds to a moving client connected via a 3G data network [19].

TABLE I
System specifications

System parameter	Value
Disk page size	4 KByte
Disk seek time	11 ms
Disk read/write rate	125 MByte/s
SCP read/write rate	80 MByte/s
SCP encryption/decryption rate	10 MByte/s
Communication bandwidth	384 Kbit/s (48 Kbyte/s)
Communication round-trip time	700ms

[3]http://www.maproom.psu.edu/dcw/

[4]http://www.seagate.com/www/en-us/products/desktops/barracuda hard drives/

[5]http://www-03.ibm.com/security/cryptocards/pcixcc/overhardware.shtml

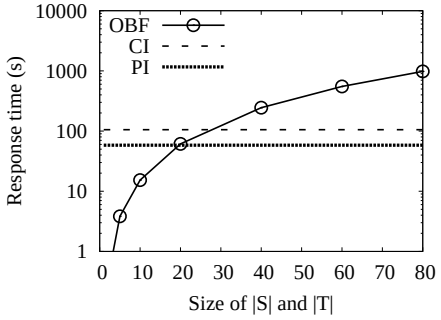

Fig. 5. Effect of $|S|$ on OBF, with $|S| = |T|$ (Argentina)

TABLE II
Components of response time (Argentina)

Method	CI	PI
Response time (s)	105.45	58.17
PIR time (s)	88.09	54.21
Communication time (s)	17.34	3.94
Client-side computations (s)	0.02	0.01
PIR page accesses of the region data file	193 of 775	2 of 775
PIR page accesses of the network index file	2 of 1,327	36 of 274,788
Total storage space (MB)	8.40	1,102

The average *response time* of a method is measured by running a workload of 1,000 shortest path queries. It denotes the elapsed time from query submission until obtaining the shortest path result. It consists of: (i) server processing time, (ii) communication time, and (iii) client-side computation time. For the obfuscation method (OBF), component (i) refers to the processing of obfuscated queries at the server. For the PIR-based methods, component (i) corresponds to the PIR time for fetching disk pages from the database.

Figure 5 illustrates the response time of the obfuscation method (OBF). To reduce the amount of information leaked (w.r.t. the original method in [12]), we form the obfuscation sets of s and t with decoys randomly and uniformly chosen in the road network, as opposed to selecting locations near them. The figure shows the overall response time versus the size of obfuscation sets S and T (where $|S| = |T|$) on the Argentina network. As expected, OBF is more efficient for small obfuscation sets. However, for $|S|$ and $|T|$ greater than 20, OBF is slower than CI and PI (whose response times are represented by horizontal lines) due to its large communication and server processing costs.

Regarding the comparison between the strong privacy methods, CI and PI, in Table II we present measurements using the same road network (Argentina). In both cases, the response time is dominated by the PIR cost, while the communication time and the client-side computations account only for a small fraction of the total time. PI is almost two times more efficient than CI. To interpret performance, the table also shows the number of PIR accesses in the region data and network index files. CI incurs 5 times more PIR accesses than PI. However,

they have a smaller difference in response time. This is because PI performs retrievals on a much larger network index file than CI, i.e., each access costs more. Table II also presents the space requirements of the schemes. PI has the largest database, due to its voluminous index.

VI. Open Questions and Conclusion

After an exposure to location privacy paradigms, there are important questions and concerns to consider. We know that LBSs are curious about the locations/queries of their clients (e.g., [1], [24] mentioned in the Introduction). Research has looked ahead, and there are solid methods (or a good start at least) for protecting location privacy. The questions, however, that will judge the significance of these contributions (and will guide future research) should be answered by the users. For example, do they care to protect their location from the LBSs or they trust them enough to let them collect these data? If the users are concerned about their privacy, to what degree are they willing to sacrifice system responsiveness in order to protect it? Would they be convinced by the (sort of ad-hoc) spatial k-anonymity model, or they demand strong privacy guarantees? In the latter case, would they tolerate two orders of magnitude longer response times (w.r.t. unprotected query processing)?

Other questions that will determine the significance of location privacy techniques is whether LBSs are willing to adopt a private query processing model. This option would disallow them gain of valuable information about their users and would incur both one-off and ongoing costs. For example, trusted hardware (e.g., SCPs) is expensive. Private processing schemes add both space and time overheads, to be born by the LBS. In the case of PI, for example, processing cost is several times higher than unprotected processing, and requires around 1 GByte storage for a network of fewer than 100,000 nodes. The only incentive for LBSs to adopt a private processing option would be if the users strongly demanded it, or by fear that competition may start offering it. A general impression is that this demand has started showing, but whether it is strong enough remains to be seen.

The research frontier could also make private processing options more appealing, by offering solutions of improved space and time overheads, or by proposing different privacy models. For shortest path queries, for example, lossless or lossy compression of network data (taking into account their characteristics/structure) could improve the performance of CI/PI. Another idea is to develop approximate schemes with bounded cost deviation from the actual shortest path. The adoption of a different model, other than PIR or obfuscation, could also lead to a convincing and more practical solution.

References

[1] J. Cheng. How Apple tracks your location without your consent and why it matters. http://arstechnica.com/apple/news/2011/04/how-apple-tracks-your-location-without-your-consent-and-why-it-matters.ars, April 2011.

[2] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS*, pages 41–50, 1995.

[3] C.-Y. Chow, M. F. Mokbel, and X. Liu. A peer-to-peer spatial cloaking algorithm for anonymous location-based service. In *GIS*, pages 171–178, 2006.

[4] M. Duckham and L. Kulik. A formal model of obfuscation and negotiation for location privacy. In *Pervasive*, pages 152–170, 2005.

[5] W. I. Gasarch. A survey on private information retrieval (column: Computational complexity). *Bulletin of the EATCS*, 82:72–107, 2004.

[6] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K.-L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD Conference*, pages 121–132, 2008.

[7] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVE: anonymous location-based queries in distributed mobile systems. In *WWW*, pages 371–380, 2007.

[8] P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE TKDE*, 19(12):1719–1733, 2007.

[9] A. Khoshgozaran and C. Shahabi. Blind evaluation of nearest neighbor queries using space transformation to preserve location privacy. In *SSTD*, pages 239–257, 2007.

[10] A. Khoshgozaran, C. Shahabi, and H. Shirani-Mehr. Location privacy: going beyond k-anonymity, cloaking and anonymizers. *Knowl. Inf. Syst.*, 26(3):435–465, 2011.

[11] H. Kido, Y. Yanagisawa, and T. Satoh. An anonymous communication technique using dummies for location-based services. In *ICPS*, pages 88–97, 2005.

[12] K. C. K. Lee, W.-C. Lee, H. V. Leong, and B. Zheng. Navigational path privacy protection: navigational path privacy protection. In *CIKM*, pages 691–700, 2009.

[13] R. H. Möhring, H. Schilling, B. Schütz, D. Wagner, and T. Willhalm. Partitioning graphs to speedup Dijkstra's algorithm. *ACM Journal of Experimental Algorithmics*, 11, 2006.

[14] M. F. Mokbel, C.-Y. Chow, and W. G. Aref. The new Casper: Query processing for location services without compromising privacy. In *VLDB*, pages 763–774, 2006.

[15] K. Mouratidis and M. L. Yiu. Anonymous query processing in road networks. *IEEE TKDE*, 22(1):2–15, 2010.

[16] K. Mouratidis and M. L. Yiu. Shortest path computation with no information leakage. *PVLDB*, 5(8):692–703, 2012.

[17] S. Papadopoulos, S. Bakiras, and D. Papadias. Nearest neighbor search with strong location privacy. *PVLDB*, 3(1):619–629, 2010.

[18] D. Riboni, L. Pareschi, and C. Bettini. Integrating identity, location, and absence privacy in context-aware retrieval of points of interest. In *MDM*, pages 135–140, 2011.

[19] P. Romirer-Maierhofer, F. Ricciato, A. D'Alconzo, R. Franzan, and W. Karner. Network-wide measurements of TCP RTT in 3G. In *TMA*, pages 17–25, 2009.

[20] R. Sion and B. Carbunar. On the practicality of private information retrieval. In *NDSS*, 2007.

[21] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[22] C. R. Vicente, I. Assent, and C. S. Jensen. Effective privacy-preserving online route planning. In *Mobile Data Management (1)*, pages 119–128, 2011.

[23] T. Wang and L. Liu. Privacy-aware mobile services over road networks. *PVLDB*, 2(1):1042–1053, 2009.

[24] C. Williams. Apple under pressure over iphone location tracking. http://www.telegraph.co.uk/technology/apple/8466357/Apple-under-pressure-over-iPhone-location-tracking.html, April 2011.

[25] P. Williams and R. Sion. Usable PIR. In *NDSS*, 2008.

[26] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis. Secure kNN computation on encrypted databases. In *SIGMOD Conference*, pages 139–152, 2009.

[27] M. L. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis. Enabling search services on outsourced private spatial data. *VLDB J.*, 19(3):363–384, 2010.

[28] M. L. Yiu, C. S. Jensen, X. Huang, and H. Lu. Spacetwist: Managing the trade-offs among location privacy, query performance, and query accuracy in mobile services. In *ICDE*, pages 366–375, 2008.